MiniLedger: Compact-sized Anonymous and Auditable Distributed Payments

P. Chatzigiannis and F. Baldimtsi

GEORGE MASON UNIVERSITY

Motivation

Crypto.IQ Market Tech Regulation Tips & Tools Opinion Login Sign Up

Opinion

BitBay Bans Monero (XMR) As Exchanges Worldwide Delist Privacy Coins

November 27, 2019 | by Crypto.IQ

Share This Post

BitBay, which is a cryptocurrency exchange based in Europe, has banned Monero (XMR). This follows Monero (XMR) being banned from other exchanges across the world, including Upbit and OKEx Korea.

Monero (XMR) is perhaps the most desirable privacy coin, since it fully anonymizes the origin, destination, and amount of a transaction. These anonymity features are mandatory on the Monero (XMR) network, unlike Zcash (ZEC) which is optionally transparent.

Although the anonymity features of Monero (XMR) may be good for users who want to maintain privacy, regulators consider this anonymity to be dangerous, since there is no way to know if Monero (XMR) is being used for money laundering or other illegal activity. Likewise, exchanges which offer Monero (XMR) are taking the risk that they are facilitating illegal activity, which is in violation of the anti-money laundering (AML) protocols required to be officially licensed.

An international council of regulators called the FATF has actually mandated that cryptocurrency exchanges worldwide ban privacy coins by 2020, and it seems most countries will enforce this rule.

Privacy Coins Monero, Zcash And Dash Face Uphill Battle In Japan

Michael del Castillo Forbes Staff
Crypto & Blockchain
I cover enterprise adoption of blockchain and cryptocurrency.

Mount Fuji, the highest mountain in Japan, from the Fujikawaguchiko, Yamanashi prefecture.
AFP VIA GETTY IMAGES

Japanese financial regulators are playing hard-ball when it comes to anonymous cryptocurrencies. Tokyo-based cryptocurrency exchange

- Lack of privacy and anonymity in decentralized ledger-based currencies
- ZCash and Monero have strong privacy guarantees for *permissionless* ledgers
- But private payment systems offer no built-in auditing mechanisms to protect from illegal transactions

Can we offer privacy while allowing for auditability?

Related Works

- Provisions (CCS15) - Proofs of solvency for bitcoin exchanges
- “Accountable Privacy for Decentralized Anonymous Payments” (FC16) - Accountability extensions proposed to Zcash - centralized authorities
- PRCash (FC19) – Permissionless, accountable w.r.t. transaction value, centralized authorities
- PGC (ESORICS20) – Permissionless, auditable, no tx graph hiding
- zkLedger (NSDI18) – Permissioned, fully anonymous, ledger grows linearly with # of participants and # transactions

Our work - MiniLedger

- Ledger-based payment system in the permissioned setting
- Fully private and auditable, without the need of centralized authorities
- Near-constant size of ledger in terms of transactions
- Inspired from zkLedger, but addressing vulnerabilities and shortcomings
- Linear in terms of participants (Banks), but can be extended to accommodate an arbitrary number clients

Building Blocks – Commitment Scheme

- Digital equivalent of a sealed box
- Hiding + binding

- Homomorphic commitments:

+

+

=

Building Blocks – NIZKs

The lockbox has $100

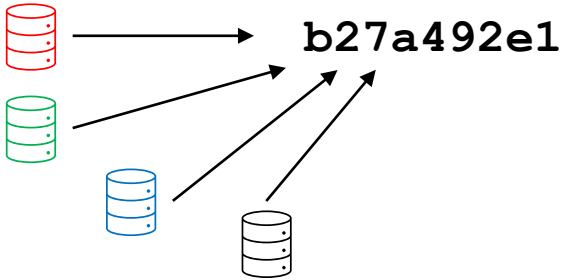
Alice knows how much money is in the box

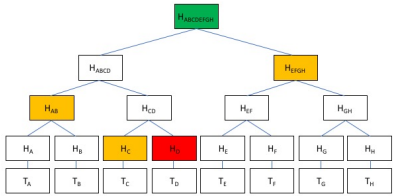
Prover

π

Verifier

Building Blocks – Cryptographic Accumulators





Prover

∈ **b27a492e1** , w

Verifier

Overview of zkLedger

Intuition for zkLedger

- Account-based payment system
- Semi-homomorphic commitments (Pedersen commitments)

	B_1	...	B_2	...	B_3
...					
tx_{15}	$cm(-\$1, r_1)$		$cm(\$1, r_2)$		$cm(\$0, r_3)$

zkLedger overview

	B_1	…	B_j		B_n
…					
tx_i			$cm_{ij}= g^v h^r$, $Token_{ij}=pk_j^r$ π^A_{ij}, π^C_{ij} cm_{ij}', $Token_{ij}'$		
…					

- cm: Pedersen commitment
- Token, Token': used in audits
- $pk = h^{sk}$
- π^A: disjunctive NIZK proof that (Bank has assets to spend and authorizes transfer) or receives funds, with range proofs
- π^B: No funds created or destroyed ($\Sigma v_i = 0$)
- π^C: NIZK proof of equality for r between cm and Token

cm': recommitment to v ***or*** to Σv_j. Used to combine range proofs in π^A into a single proof.
v>0: Bank receives
v<0: Bank spends
v=0: Bank does not participate in transaction

Audits in zkLedger

	B_1	…	B_j	…	B_n
tx_1	…	…	cm, Token, π^A, π^C, cm', Token'	…	…
tx_2	…	…	cm, Token, π^A, π^C, cm', Token'	…	…
tx_3	…	…	cm, Token, π^A, π^C, cm', Token'	…	…

Auditor

$?

$\Sigma v, \pi^{Aud}$

Bank j

- Auditor asks for the value of the Bank's total assets
- Bank replies with a value and a ZKP π^{Aud} : $\prod cm_j/g_j^v = \prod Token_j^{sk}$
- Can audit single transaction or do more complex audits (e.g. statistical queries)

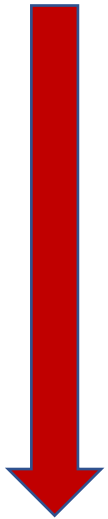
Vulnerabilities – shortcomings in zkLedger

- "Unknown value" attack:
  - zkLedger mechanism for unknown randomness (Token)
  - But no mechanism for unknown values..
  - Affected Bank won't be able to pass audits
- Out of band communication → Why not open commitment?
- "Always online" assumption defeats whole scheme and trivializes it!

Fix: semi-homomorphic encrypt instead of commit

	B_1	...	B_2	...	B_3
...					
tx_{15}	$\mathsf{cm}(-\$1, r_1), \mathsf{Token}(pk, r_1)$		$\mathsf{cm}(\$1, r_2), \mathsf{Token}(pk, r_2)$		$\mathsf{cm}(\$0, r_3), \mathsf{Token}(pk, r_3)$

Linear storage of public ledger in the number of transactions (maintained by everyone!)



	B_1	...	B_j	...	B_n
...					
tx_i			$cm_{ij} = g^{v_{ij}} h^{r_{ij}}$ $Token_{ij} = pk_j{}^{r_{ij}}$ $\pi^A_{ij}, \pi^C_{ij}, cm'_{ij}, Token'_{ij}$		
...					
...					
...					
...					
...					
...					

- A ledger with 100 Banks and just 5000 transactions needs 32GB of storage!
- Computational performance also degrades

MiniLedger

Goal: Private, auditable transactions in the permissioned setting *without* public linear storage requirements

High level idea:

- Consensus: agreement among participating parties
- Compact Data Structures

MiniLedger main building blocks

- Pedersen commitments
- NIZK proofs (Σ-protocols) + AND-OR compositions
- Ledger Consensus
- Accumulators (additive – positive or universal – trapdoorless – no upper bound)
- Additive ElGamal encryption variant

Additive ElGamal variant

Normal additive ElGamal: pp: (G, g, p)

$pk = g^{sk}$ $(c_1, c_2) = (g^r, g^m pk^r)$

ElGamal variant: pp: $(G, g, \mathbf{h}, p)$

$pk = \mathbf{h}^{sk}$ $(c_1, c_2) = (\mathbf{pk}^r, g^m \mathbf{h}^r)$

Dec: $g^m = c_2 / c_1^{1/sk} \rightarrow$ recover m from lookup table

- Variant can homomorphically add ciphertexts c_2 generated under different public keys
- $c_2 \times c_2'$ contains encryption of $(m + m') \rightarrow$ if $c_2 \times c_2' = 1$ then $m = -m'$

Assumptions (similar to zkLedger)

- Network: no eclipse attacks, transactions sent using anonymous broadcast
- Consensus: no dishonest majority, no forks
- No “race conditions” when creating transactions
- Set of Banks is static (dynamic set can trivially still be supported)
- Incentives orthogonal to our setting
- RO model for NIZKs

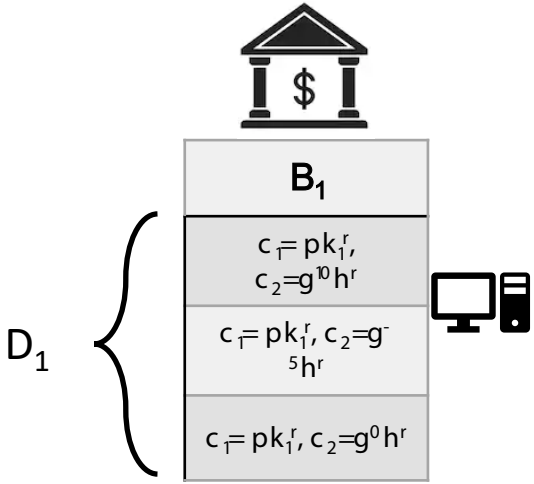
MiniLedger overview

	B_1	…	B_n
tx_1	$c_1 = pk_1^r, c_2 = g^{10}h^r$ π^A, π^C, cm		…

- ElGamal variant is equivalent to Pedersen commitment + Token
- π^C : ensures correct decryption
- No extra Token'

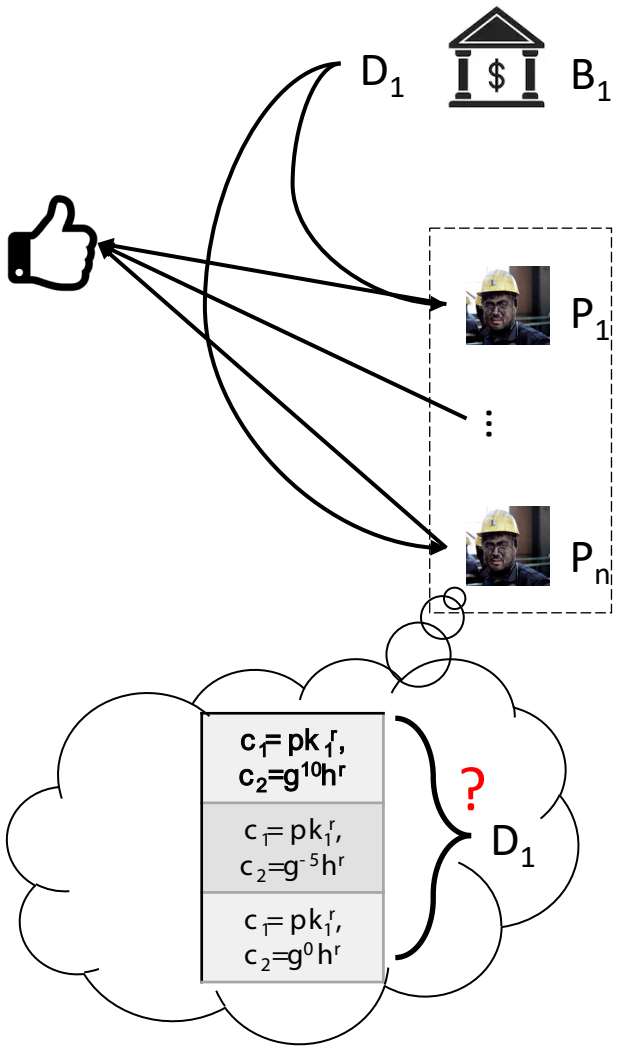
Transaction pruning

	B_1	…	B_n
tx_1	$c_1 = pk_1^r, c_2 = g^{10} h^r$ π^A, π^C, cm		…
tx_2	$c_1 = pk_1^r, c_2 = g^{-5} h^r$ π^A, π^C, cm		$c_1 = pk_1^r, c_2 = g^5 h^r$ π^A, π^C, cm
tx_3	$c_1 = pk_1^r, c_2 = g^0 h^r$ π^A, π^C, cm		…



Transaction pruning

	B_1	…	B_n
tx_1	$c_1 = pk_1^r, c_2 = g^{10}h^r$ π^A, π^C, cm		…
tx_2	$c_1 = pk_1^r, c_2 = g^{-5}h^r$ π^A, π^C, cm		$c_1 = pk_1^r,$ $c_2 = g^5h^r$ π^A, π^C, cm
tx_3	$c_1 = pk_1^r, c_2 = g^0h^r$ π^A, π^C, cm		…
	D_1		



Transaction pruning

	B_1	…	B_n
tx_1			…
tx_2			$c_1 = pk_1^r$, $c_2 = g^5 h^r$ π^A, π^C, cm
tx_3			…
	D_1		

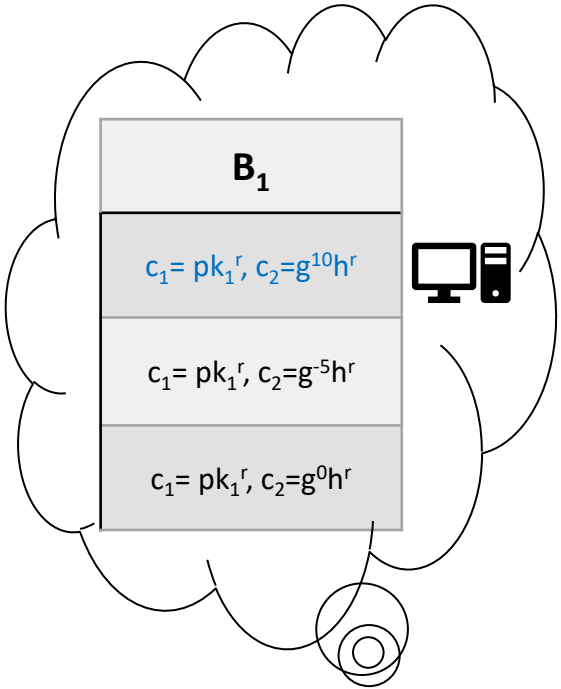
D_1 includes $cm = \Pi c_2$

Ideal Public Ledger

B_1	…	B_n
D_1		D_n

Audits on pruned transactions

	B_1	...	B_n
tx_1			...
tx_2			...
tx_3			...
	D_1		



$? for tx_1

$(1||c_1||c_2)$, w, 10, π^{Aud}

Concrete efficiency improvements vs zkLedger

- Token' is redundant → reduced ledger storage cost
- Combine NIZKs π_A, π_C → reduce proof size and computation cost

		Prime-order exps	Prime-order multi-exps	Exponents
Transaction cost	zkLedger	$5n$	$8n$	-
	MiniLedger	$4n$	$7n$	-
Verification cost	zkLedger	$12n$	$6n$	-
	MiniLedger	$11n$	$5n$	-
Storage cost	zkLedger	$5n$	$8n$	$12n$
	MiniLedger	$4n$	$7n$	$10n$

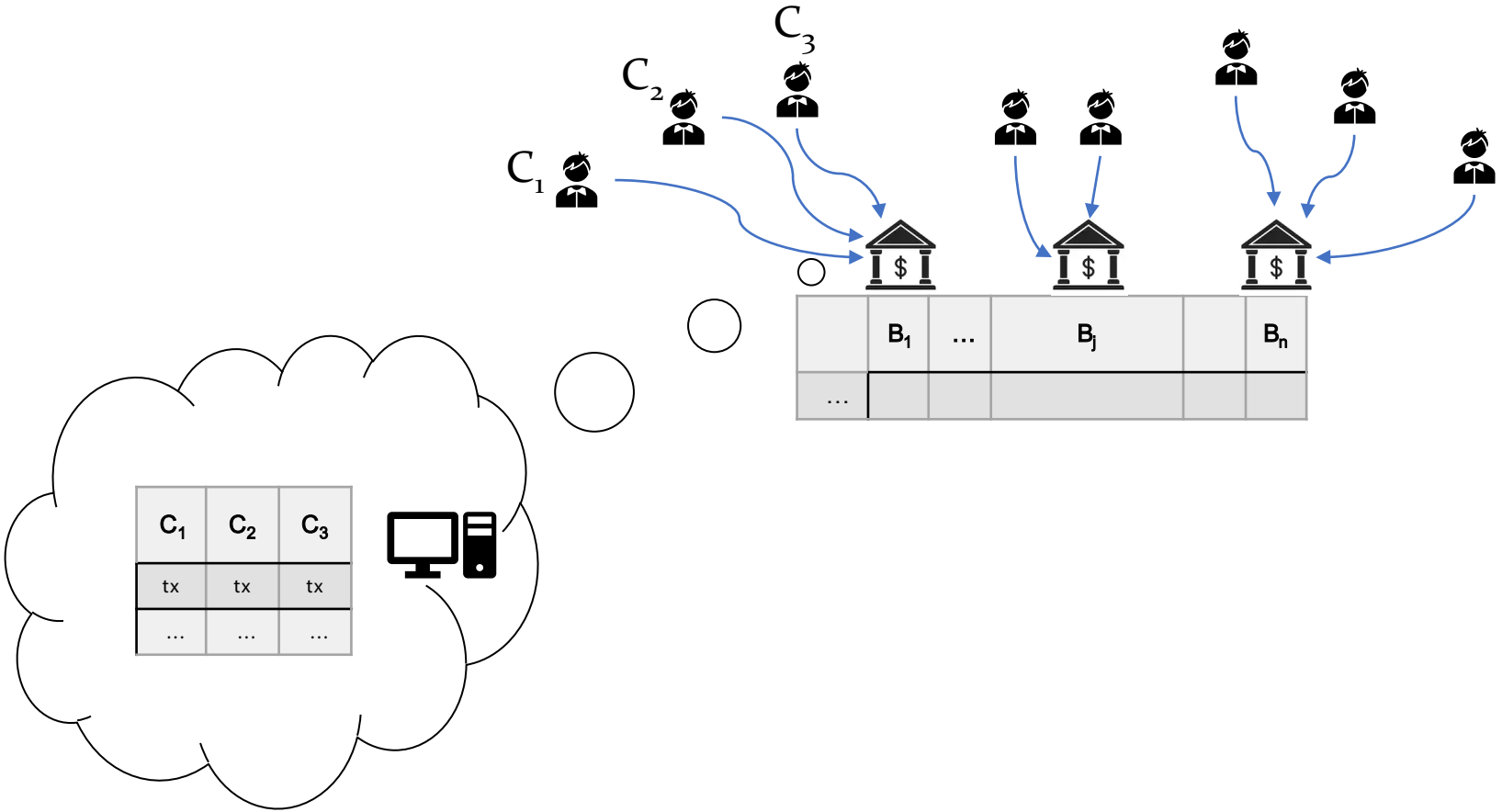
Asymptotic efficiency improvements vs zkLedger

- zkLedger: O(mn), n: #Banks, m: #txs
- miniLedger: O(n)

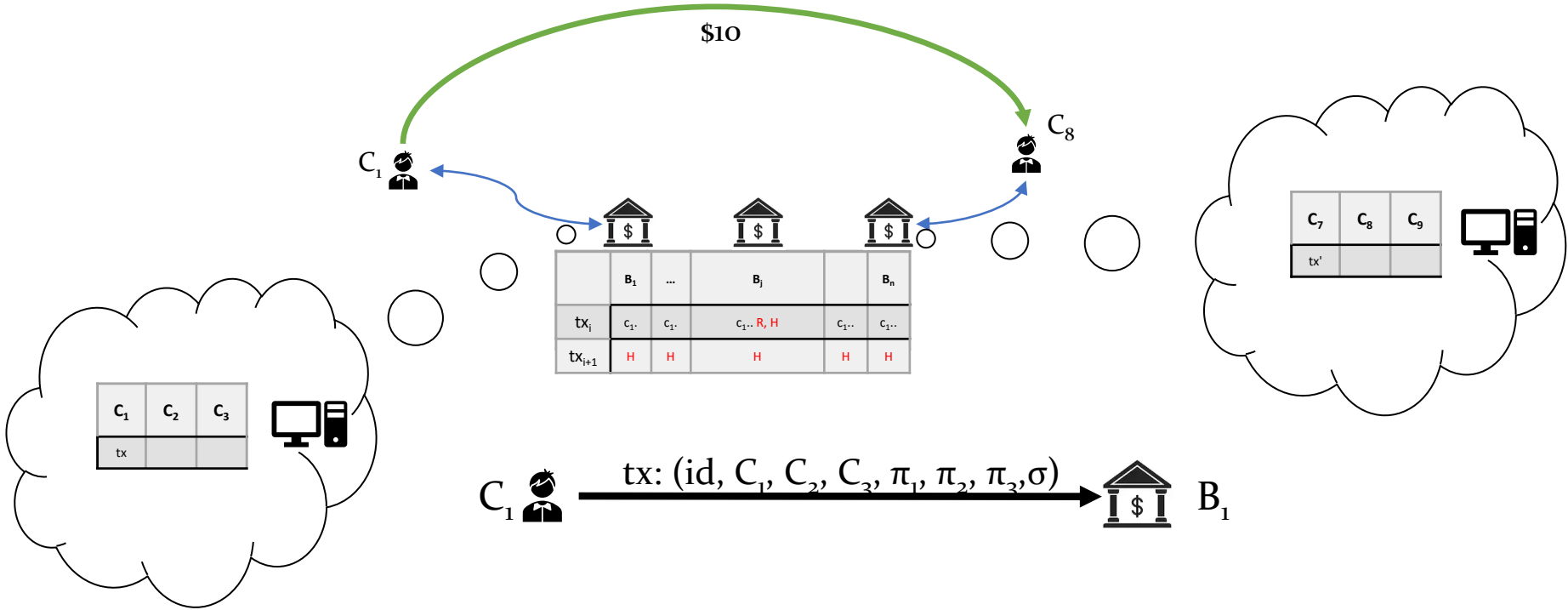
MiniLedger Extensions

- Audits without consent (→ non-interactive)
  - Enforce inclusion of NIZK π^{Aud} in the output of a transaction - encrypt claimed value under trusted auditor's pk
- Trusted auditor can be substituted by a threshold encryption protocol (t out of n Banks can audit)

Adding clients for fine-grained auditing



Client to client transactions



Auditing users and aggregating transactions

- Banks can aggregate many of their clients' transactions into a single MiniLedger transaction
- Auditors reactively check Banks' honest behavior
  - Auditor requests private table from Bank
  - Check that sum of private table always matches sum of assets on MiniLedger
  - Audits in a client level similar to MiniLedger
  - Bank is accountable for any inconsistencies between private table and main Ledger
- Extension maintains indistinguishability properties of the ledger, fully compatible with pruning

Additional audit types

- Transaction exceeding some amount (e.g. v > $10K)
- Client exceeding threshold over a period of time
- Has sent assets to a specific client (proofs on non-membership)

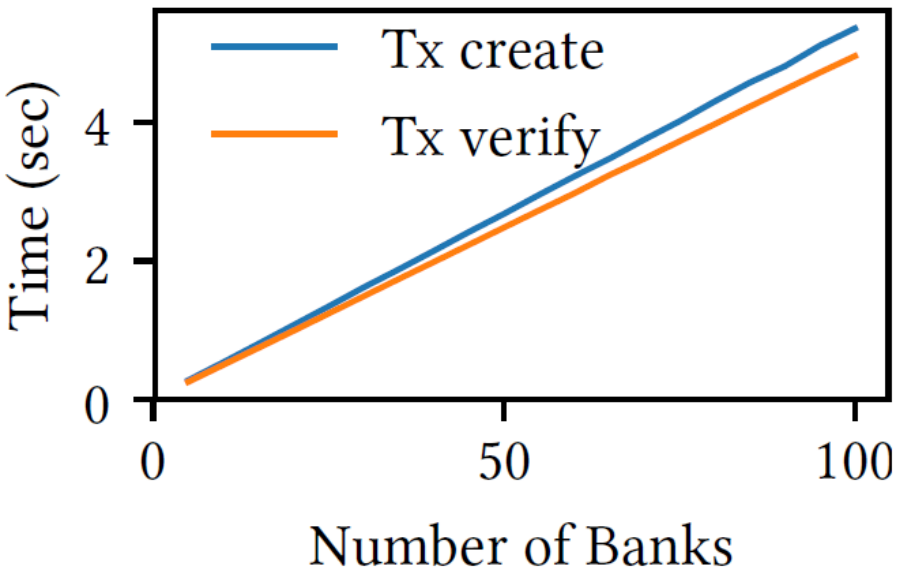
Implementation

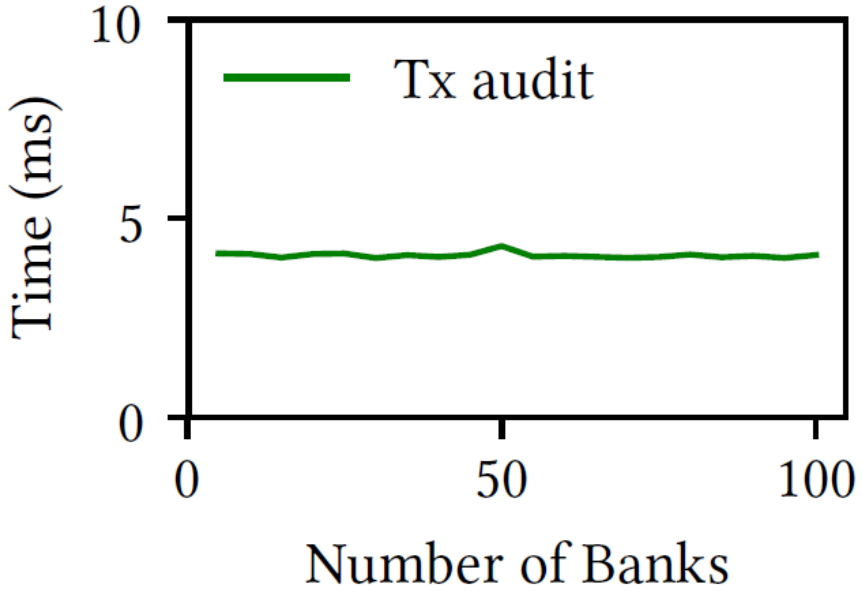
- Prototype in python using petlib (secp256k1 curve) and zksk (Schoenmakers range proofs) libraries
- Bulletproof compatible → further optimizations possible
- Implementation options for Accumulator (depending on use-case):
  - Merkle Trees
  - Batched-RSA accumulators

Merkle Trees and batched-RSA accumulators

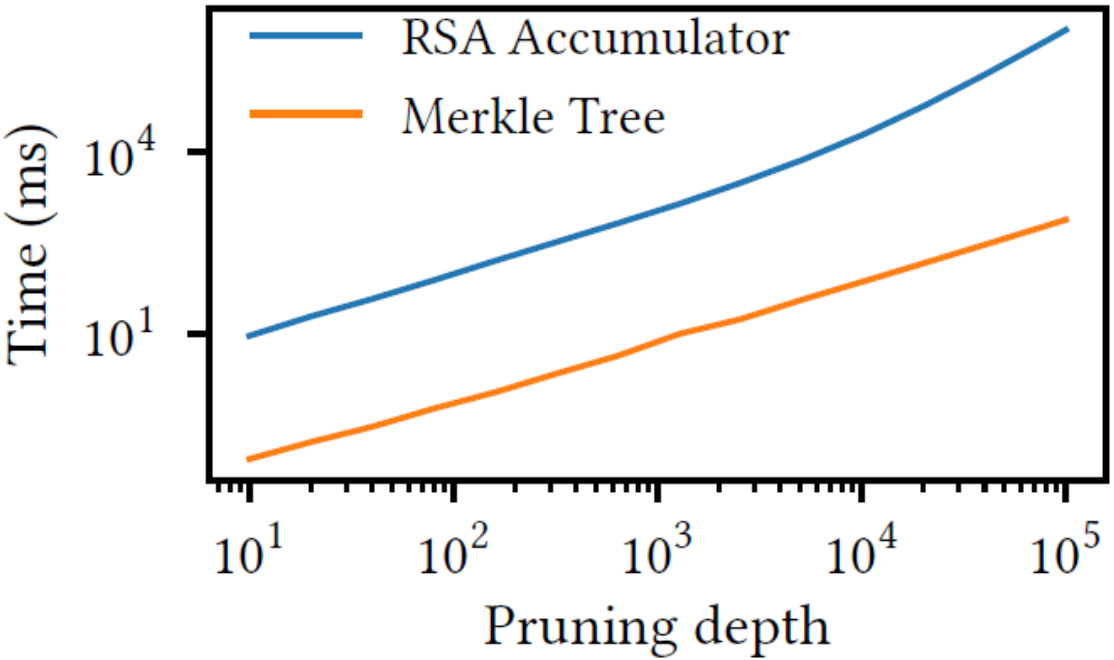
- Merkle Trees: straightforward, naturally support order binding, cheap Merkle proofs but over-linear complexity for many transactions, no support for non-membership proofs
- RSA Accumulators: need a deterministic prime mapping function, trapdoor can be computed (and later discarded) by a TP or MPC, more efficient in large “batch” operations (e.g. auditing sums)
- VC more expensive overall and offer unneeded properties (cannot commit to same index)

Evaluation – Transaction creation, verification and auditing

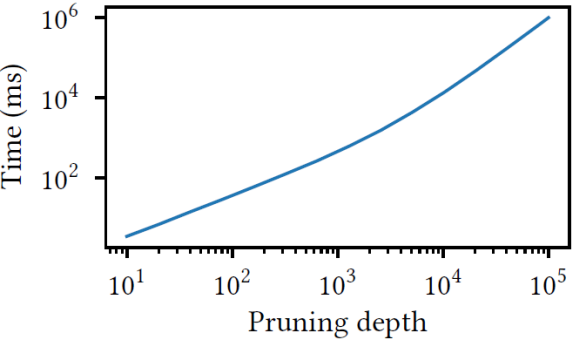




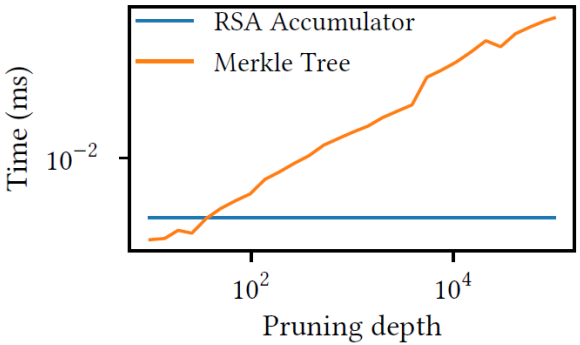
Evaluation overview – Pruning operation



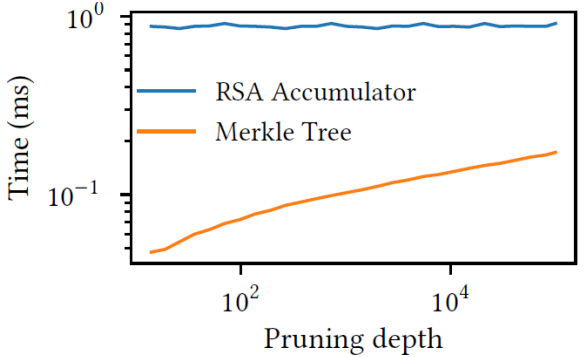
Evaluation overview – Single transaction audit



RSA Wit Generation cost

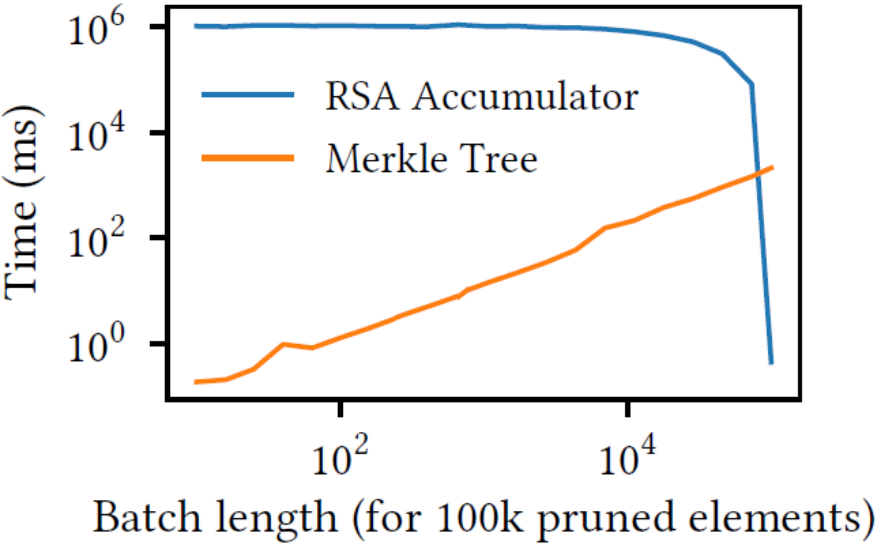


Audit open cost for one tx

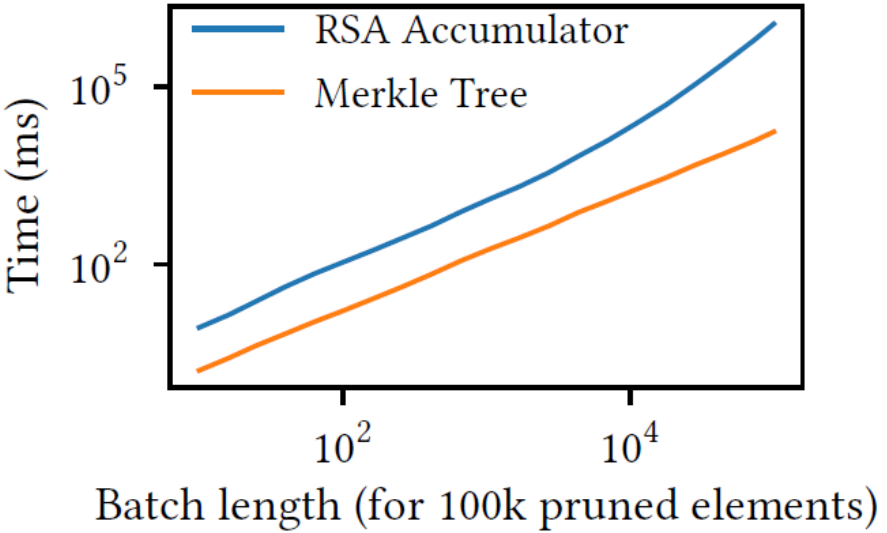


Audit verify cost

Evaluation overview – Multi transaction audit



Batch audit open costs



Batch audit verify costs

Evaluation highlights

- zkLedger size 512nm bits, MiniLedger lower bound at 256n bits (all n Banks have pruned, single-line Ledger)
- Same computation costs for a single transaction creation/verification (without optimizations)
- Pruning costs independent to # of banks
- Pruning 100k txs need ~1.5 minutes with RSA, ~1ms for Merkle Trees
- Auditing costs same with zkLedger if transaction is not pruned
- Additional costs when transaction is pruned (as in previous Figures)

Conclusions

- Based on zkLedger, we construct an auditable payment system without linear public storage requirements
- Choice of compact data structure is based on use-case
- Given a (nearly) constant public ledger, we can add clients for fine-grained auditability and accommodate a large user base.

Questions?